Dlopen()

Vulnerable to TOCTOU issues

Sean Barnum, Cigital, Inc. [vita[1]]

2007-03-22

Part "Original Cigital Coding Rule in XML"

Mime-type: text/xml, size: 5562 bytes

Attack Category	• Path spoofing or confusion problem
Vulnerability Category	• Indeterminate File/Path • TOCTOU - Time of Check, Time of Use
Software Context	• Filename Management
Location	• dlfcn.h
Description	Care must be exercised when accessing files from passed in pathnames. dlopen(const char *file, int mode) takes a filename (or pathname) that can potentially be subjected to a time-of-check-to-time-of-use race condition during an access control check. dlopen() creates an executable object (e.g. shared library, relocatable library, programs) file specified by the file argument. dlopen() is vulnerable to changes to or symbolic linking of the target filename allowing the creation of a nonintended executable object.
APIs	**Function Name** | **Comments** dlopen |
Method of Attack	The key issue with respect to TOCTOU vulnerabilities is that programs make assumptions about atomicity of actions. It is assumed that checking the state or identity of a targeted resource followed by an action on that resource is all one action. In reality, there is a period of time between the check and the use that allows either an attacker to intentionally or another interleaved process or thread to unintentionally change the state of the targeted resource and yield unexpected and undesired results An attacker could link the target file to a known executable object file and trick the system into creating the attackers own executable object.
Exception Criteria	

1. http://buildsecurityin.us-cert.gov/bsi/about_us/authors/35-BSI.html (Barnum, Sean)

Solutions

Solution Applicability	Solution Description	Solution Efficacy
Generally applies.	The most basic advice for TOCTOU vulnerabilities is to not perform a check before the use. This does not resolve the underlying issue of the execution of a function on a resource whose state and identity cannot be assured, but it does help to limit the false sense of security given by the check.	Does not resolve the underlying vulnerability but limits the false sense of security given by the check.
Generally applies.	Limit the interleaving of operations on files from multiple processes.	Does not eliminate the underlying vulnerability but can help make it more difficult to exploit.
Generally applies.	Limit the spread of time (cycles) between the check and use of a resource.	Does not eliminate the underlying vulnerability but can help make it more difficult to exploit.
Generally applies.	Recheck the resource after the use call to verify that the action was taken appropriately.	Checking the filename permissions after the operation does not change the fact that the operation may have been exploited but it does allow

		halting of the application in an error state to help limit further damage.
When user specification of the file to be altered is not necessary.	Do not rely on user-specified input to determine what the file to be made executable.	This will reduce exposure but will not eliminate the problem.

Signature Details	

Examples of Incorrect Code	`#include`
	`#include`
	
	`/* check permissions to the file*/`
	`if(!access(file, ...){`
	`/* make the file executable*/`
	`dlopen(file, ...)`
	`}`
	`else{`
	`/* permission was denied */`
	`}`

Examples of Corrected Code	
Source References	• http://www.opengroup.org/onlinepubs/009695399/functions/dlopen.html • Viega, John & McGraw, Gary. *Building Secure Software: How to Avoid Security Problems the Right Way*. Boston, MA: Addison-Wesley Professional, 2001, ISBN: 020172152X • ITS4 Source Code Vulnerability Scanning Tool[3]
Recommended Resource	
Discriminant Set	**Operating System**: • UNIX **Languages**: • C • C++

Cigital, Inc. Copyright

1. mailto:copyright@cigital.com